

# SDDP vs. ADP: The Effect of Dimensionality in Multistage Stochastic Optimization for Grid Level Energy Storage

Tsvetan Asamov, Daniel F. Salas and Warren B. Powell

Department of Operations Research and Financial Engineering, Princeton University  
 {tasamov, dsalas, powell}@princeton.edu

There has been widespread interest in the use of grid-level storage to handle the variability from increasing penetrations of wind and solar energy. This problem setting requires optimizing energy storage and release decisions for anywhere from a half-dozen, to potentially hundreds of storage devices spread around the grid as new technologies evolve. We approach this problem using two competing algorithmic strategies. The first, developed within the stochastic programming literature, is stochastic dual dynamic programming (SDDP) which uses Benders decomposition to create a multidimensional value function approximations, which have been widely used to manage hydro reservoirs. The second approach, which has evolved using the language of approximate dynamic programming, uses separable, piecewise linear value function approximations, a method which has been successfully applied to high-dimensional fleet management problems. This paper brings these two approaches together using a common notational system, and contrasts the algorithmic strategies (which are both a form of approximate dynamic programming) used by each approach. The methods are then subjected to rigorous testing using the context of optimizing grid level storage.

*Key words:* multistage stochastic optimization, approximate dynamic programming, energy storage, stochastic dual dynamic programming, Benders decomposition

*History:*

## 1. Introduction

There is global interest in increasing the generation of electricity from renewables to meet the pressure to reduce our carbon footprint. However, wind and solar energy cannot be directly controlled (they are not “dispatchable” in the parlance of the energy systems community), and in addition to their predictable variability (e.g. the rising and setting of the sun), they introduce a high level of uncertainty. Over the years, grid operators have developed a sophisticated planning process to plan power needs in the presence of modest levels of uncertainty, due primarily to changes in weather. There is a growing consensus that storage will be needed to smooth the variations introduced by wind and solar, which requires making decisions about when and where to charge and discharge

batteries (or other storage devices). Storage devices will need to be managed in a way that captures both the effects of grid congestion, as well as the real-time control of generators.

The process of managing energy generation and transmission by grid operators in the U.S. consists primarily of three steps: 1) the day-ahead unit-commitment problem, which determines which steam-generating units will be turned on and off (and when); 2) intermediate-term planning (typically every 15 to 30 minutes) which determines which gas turbines will be turned on and off, and 3) real-time economic dispatch (every 5 minutes), which is the process of increasing/decreasing (“ramping”) the output of generators (both steam and gas turbines) in response to current conditions. To handle unexpected variations (due to weather and load variations which are caused in part by the sun), grid operators plan reserve capacity, typically in two forms: spinning reserve, which can be tapped immediately, or nonspinning reserve, which is usually gas turbines that can be brought online in a few minutes.

This process handles unexpected variations, but not at the level that would be experienced at the high penetrations of renewables that are being targeted by state renewable portfolio standards in the U.S., as well as national targets being set by nations around the world. As grid operators draw close to 20 percent renewables from wind and solar (in Germany it is over 30 percent), it is widely anticipated that grid-level storage will be needed to help smooth the variations to meet grid capacity constraints and to handle the gaps between the rate of change from wind and solar, and the ramping ability of online generators.

Energy storage is an emerging technology at this time. A major grid such as that operated by PJM may have only 5-10 storage devices (batteries and pumped hydro) but the number is growing quickly as the technology improves and costs come down. In addition to the possibility of batteries being installed in individual communities to help with outages, vehicle-to-grid technology already exists that allows an energy aggregator to treat a few hundred cars as a single storage “device.” It is easy to envision a grid with hundreds of large storage devices, and perhaps thousands of smaller ones.

This paper addresses the algorithmic challenge of simultaneously optimizing decisions to charge and discharge energy in a network of grid-connected batteries, while also optimizing ramping decisions at each generator that is currently operating. These decisions have to also respect transmission constraints. We wish to test this logic at different levels of investment in solar generation capacity. To do this, we first use a model called SMART-ISO which simulates the day-ahead, intermediate and real-time planning processes at PJM; this model was carefully calibrated against historical

performance at PJM and was shown to accurately replicate network behavior Simão et al. (2015). However, SMART-ISO, which takes approximately 2-3 hours to simulate a single week, is unable to optimize storage. For this reason, we use the unit commitment decisions from SMART-ISO (which adapts to any level of solar investment we would like to simulate), but then use a model that optimizes only the ramping decisions along with grid congestion, while also optimizing charge and discharge decisions for storage devices.

A method for optimizing grid-level storage was recently proposed in Salas and Powell (2015) based on approximate dynamic programming. This method uses separable, piecewise linear value functions to capture the value stored in each device around the grid, at 5-minute increments. The method was shown to produce near-optimal solutions for deterministic problems, which was the only benchmark available at the time. This leaves open the very real question of how well this methodology would work under more realistic stochastic conditions.

Far more popular in the stochastic programming community has been the use of Benders decomposition, primarily in the context of a methodology known as stochastic dual dynamic programming (SDDP), which was first proposed in Pinto (1991) for the management of water reservoirs. This has produced a flurry of follow-on papers (Mo and Gjelsvik (2001), Shapiro (2011), Lohndorf et al. (2013)) focusing primarily on applications in the planning of hydroelectric power, as well as considerable theoretical interest (Linowsky and Philpott (2005), Philpott and Guan (2008), Shapiro et al. (2014), Girardeau and Philpott (2015)). SDDP has generally made the assumption of “intertemporal independence” which is that new information becoming available at time  $t$  does not depend on the history of the process. Benders has been used in conjunction with scenario trees that capture the history of the information process (Birge (1985), Sen and Zhou (2014)), but this work has not proven to be computationally tractable. Further, even with the assumption of intertemporal independence, SDDP can only be applied to a sampled approximation. Researchers have argued that the use of a sampled model produces small errors (see e.g. Shapiro et al. (2014)), but we are unaware of any research evaluating errors in specific decisions, such as the congestion on a transmission line that might guide hundreds of millions of dollars in new investment. Finally, it is generally well known that Benders struggles with high dimensional problems, where “high” might mean more than 20 storage devices. However, we are unaware of any formal study of errors that arise when using Benders (or separable approximations) as a function of the dimensionality of the resource vector. This is a question we address in this paper.

Recent research has mitigated some of the limitations of classical SDDP. Asamov and Powell (2015b) presented a version of Benders decomposition with a new regularization strategy designed for multistage problems (regularization has long been recognized as a powerful technique for Benders, but this is the first extension to multiperiod problems). Further, this work also considers a version that exhibits a first-order Markov information process.

This paper, then, uses the setting of grid level storage on the PJM grid to compare two algorithmic strategies:

- 1) Regularized Benders decomposition for multiperiod (and multistage) problems (SDDP), with independent and first-order Markov information processes. This method can only be run on a sampled information process.
- 2) Approximate dynamic programming using separable, piecewise linear value function approximations (ADP-SPWL). This method can also be run using independent and first-order Markov information processes, but is not limited to a sampled version of the problem.

These experiments will provide the first serious benchmark for both algorithmic strategies. Benders decomposition has long enjoyed bounds, but for our problem, we show that these bounds are not very tight, and further provide little insight into the accuracy of individual decisions which may affect investment decisions in the grid. ADP using piecewise linear value functions has been studied over the years in transportation applications (e.g. Topaloglu and Powell (2006a)), but in the past the only benchmarking has been against deterministic solutions. We would also argue that up to now, Benders decomposition has not faced serious algorithmic competition.

This paper makes the following contributions: 1) We present two algorithmic strategies, SDDP (from stochastic programming) and ADP-SPWL (from approximate dynamic programming) and show that these are both forms of approximate dynamic programming where the primary difference is how the value functions are being approximated (multidimensional Benders cuts vs. separable, piecewise linear approximations). We then highlight other differences that result specifically because of the nature of the two approximation strategies. 2) We then use a model of the PJM power grid and real-time energy generation to optimize across storage portfolios ranging from 5 to 100 batteries, which allows us to test the quality of the solution from each algorithmic strategy as a function of the dimensionality of the resource state variable. This appears to be the first comprehensive evaluation of SDDP over a wide range of dimensions, and the first formal evaluation of the ADP approach on a multidimensional stochastic problem, using the context of energy storage that introduces much higher dimensionality than has appeared in other experiments. In addition,

our problem setting involves more time periods than are typically considered (288, representing 24 hours in 5-minute increments), for a problem that is highly time-dependent. 3) We use the ADP strategy to provide the first evaluation of errors introduced in SDDP by solving a sampled model, focusing on the quality of individual decisions.

The paper is organized as follows. Section 2 provides a mathematical model of the grid-level storage problem. Section 3 presents canonical models of multistage stochastic programs, and dynamic programs and demonstrates that SDDP and ADP share the same structure, with just minor (but important) differences in approximation strategies. Section 4 provides a thorough set of comparisons between SDDP and ADP-SPWL in an energy storage setting, where we can vary the dimension of the resource state variable from 5 to 100, representing the first serious test of both algorithmic strategies over a wide range of dimensionalities. Section 5 concludes the paper.

## 2. Mathematical model of grid-level storage

Our intent is to study storage in the presence of high levels of energy from off-shore wind farms, derived as a part of a larger study of off-shore wind. The careful modeling of offshore wind is given in Archer et al. (2015). We then used a large-scale model of the PJM grid and energy markets called SMART-ISO to plan the correct level of energy from slow and fast fossil generating plants for a given level of wind penetration (see Simão et al. (2015) for a thorough description of this study). SMART-ISO models the nested planning process consisting of day-ahead, intermediate (roughly hour-ahead) and real-time planning, closely matching PJM’s actual planning process. SMART-ISO carefully replicates the uncertainty in each planning step. For example, forecast errors in planning day-ahead and hour-ahead energy from wind are based on actual errors from the forecasts that PJM uses for its planning of its own (onshore) wind farms.

SMART-ISO is a large-scale simulator of the unit commitment process that is unable to optimize storage. For this reason, we would run SMART-ISO assuming a particular investment in wind generation capacity. In our setup, SMART-ISO models power plants of various types with a total of 825 generators and a combined maximum capacity of 129,638 MW. We consider 396 gas turbines (23,309 MW), 50 combined cycle generators (21,248 MW), 264 steam generators (73,374 MW), 31 nuclear reactors (31,086 MW), and 84 conventional hydro power generators (2,217 MW). This model would then determine which generators were on or off at any given point in time. We then passed these on/off decisions to the storage model which would then optimize energy storage decisions in the presence of grid congestion. Optionally, our storage model is also allowed

to optimize ramping decisions of fossil generators (without turning them on or off), although we had to turn off this feature for some of our algorithmic testing.

Below, we present our mathematical model of the grid level storage problem, spanning storage, transmission, energy generation from fossils, and the exogenous generation of energy from wind.

### 2.1. Problem Description

We consider the grid of PJM Interconnection (or simply PJM), a large independent regional transmission operator serving the mid-Atlantic states out to Chicago. The PJM network comprises more than 9,000 buses and 14,000 transmission lines. A map of the the geographical territory that is served by PJM is shown in Figure 1 in the online supplement.

#### The Grid

We model the PJM grid as a graph  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ . The set of nodes  $\mathcal{N}$  correspond to buses in the grid and the set of edges  $\mathcal{E}$  correspond to transmission lines. More specifically, each edge  $(n_i, n_j) \in \mathcal{E}$  corresponds to the transmission line connecting bus  $n_i \in \mathcal{N}$  to bus  $n_j \in \mathcal{N}$ . Furthermore, we use the DC power flow approximation to model the power flow between buses in the grid.

#### Parameters

The set of parameters for each generator includes its power capacity and generation cost. Each storage device is characterized by its minimum and maximum energy capacity, its charging and discharging efficiency, and its variable storage cost. To present a formal description of our model, we introduce the following notation:

$\mathcal{G}$  = The set of fossil generators,

$\mathcal{B}$  = The set of storage devices,

$\mathcal{Q}$  = The set of wind farms,

$\kappa_g^l, \kappa_g^u$  = The minimum and maximum power capacities for electricity generator  $g \in \mathcal{G}$ ,

$\kappa_b^l, \kappa_b^u$  = The minimum and maximum energy capacities for storage devices  $b \in \mathcal{B}$ ,

$\eta_b^+, \eta_b^-$  = The charging and discharging multipliers (efficiencies) for each storage device  $b \in \mathcal{B}$ ,

$c_{t,g}^G$  = The vector of variable generation cost in \$/MWh for the set of generators  $g \in \mathcal{G}$  at time  $t$ ,

- $c_{t,b}^B$  = The vector of variable storage cost in \$/MWh for the set of storage devices  $b \in \mathcal{B}$  at time  $t$ ,
- $d_t$  = The vector of electricity demands (loads), in MW, for each node at time  $t = 0, \dots, T$ . We assume that electricity demand evolves deterministically,
- $\mathcal{Y}$  = The closed and convex set describing the model for the DC power flow in the grid following Kirchhoff's laws. It takes into account the structure of the electrical grid, capacities of the transmission lines and bounds on phase angles (similar to voltage limits in AC power flow)

### State variables

We let  $S_t$  be the (pre-decision) state capturing all the information we need at time  $t$  to model the system from time  $t$  onward, and we let  $S_t^x$  be the post-decision state, which is the information we need after we have made a decision  $x_t$  at time  $t$ . The evolution of states, decisions and information is described by time:

$$S_0 \xrightarrow{x_0} S_0^x \xrightarrow{\omega_1} S_1 \xrightarrow{x_1} S_1^x \xrightarrow{\omega_2} \dots \xrightarrow{\omega_T} S_T \xrightarrow{x_T} S_T^x.$$

Formally, we define the elements of the state of the system  $S_t = (R_t, I_t)$  as consisting of resource state variables  $R_t$  and (exogenous) information state variables  $I_t$ . The resource state  $R_t$  of the system at time  $t$  is a real vector that can be partitioned into subvectors as  $R_t = (R_t^B, R_t^G)$  where:

- $R_{t,b}^B$  is the amount of energy available in storage device  $b \in \mathcal{B}$  at time  $t$ ,
- $R_{t,g}^G$  is the level of power output from generators at time  $t$ . Since generators have up- and down- ramping limits, the power output at time  $t$  affects the range of feasible power output levels at time  $t + 1$ .

Please note that the power output levels  $R_t^G$  obey the on/off generator status determined by SMART-ISO, which is given by the vector  $Z^G$  defined below.

The information state  $I_t$  includes the following variables:

- $E_t^W$  = The energy from wind at time  $t$ ,
- $L_{ti}$  = The load at time  $t$  at node  $i$ ,
- $Z_{tg}^G$  = The binary variable indicating whether generator  $g \in \mathcal{G}$  is on or off (determined by SMART-ISO) during time period  $t$ .

We let  $L_t$  be the vector of loads and  $Z_t^G$  be the vector indicating which generators are turned on or off. Thus our information state is a the following vector

$$I_t = (E_t^W, L_t, Z_t^G),$$

where  $Z_t^G$  is provided exogenously, and  $R_t^G = 0$ , if  $Z_t^G = 0$  (the generator output is set to 0 when it is turned off).

We also represent the post-decision state of the system as  $S_t^x = (R_t^x, I_t^x)$  which is the state of our system immediately after a decision is made.

The post-decision resource state  $R_t^x = (R_t^{B,x}, R_t^{G,x})$  is given by the post-decision amount of energy available in storage devices, and the post-decision level of power output from generators  $R_t^{G,x}$ . Using the decision notation introduced below, we define the post-decision resource state as the pre-decision resource state  $R_t^B$  adjusted for battery inflows and outflows,

$$R_{t,b}^{B,x} = R_{t,b}^B + \eta_b^- x_{t,b}^{B-} - \eta_t^+ x_{t,b}^{B+}.$$

When modeling batteries, there is no exogenous change to the resource level, which means that the pre-decision state is given by

$$R_{t+1}^B = R_t^{B,x}.$$

If we model a water reservoir, we could account for exogenous rainfall (say, to reservoir  $b$ ) using

$$R_{t+1,b}^B = R_{t,b}^{B,x} + \hat{R}_{t+1,b}$$

where  $\hat{R}_{t+1,b}$  would be the stochastic rainfall occurring between  $t$  and  $t + 1$ .

### Decisions

Decisions are made at each time step  $t \in \{0, 1, \dots, T\}$  to determine the energy flow between the electric grid, wind farms and storage devices subject to meeting electricity demand. Energy from wind farms can be used to satisfy the current demand, or it can be transmitted directly into the storage devices. At any time period  $t$ , energy available in storage devices can be sold to satisfy the grid demand. Furthermore, energy can also be bought from the grid and transferred into storage for later use.



We partition the decision vector  $x_t \in \mathcal{X}_t$  as

$$x_t = \begin{pmatrix} x_t^{G+} \\ x_t^{B+} \\ x_t^{B-} \\ y_t \end{pmatrix},$$

where  $x_t^{G+} \in \mathbb{R}^{|\mathcal{G}|}$ , while  $x_t^{B+}, x_t^{B-} \in \mathbb{R}^{|\mathcal{B}|}$  and  $y_t \in Y$ . The vectors  $x_t^{B+}, x_t^{G+}$  denote the respective amounts of power injected into the grid by storage devices and generators, and  $x_t^{B-}$  denotes the power withdrawn from the grid by storage devices at time  $t$ .

Decisions have to reflect a number of constraints, which we describe next.

- When a generator  $g \in \mathcal{G}$  first comes online ( $(Z_{t,g}^G = 1) \cap (t = 0 \cup Z_{t-1,g}^G = 0)$ ), its power output is set to its minimum power capacity. Thus we have the initial generation constraints:

$$\begin{aligned} x_{0,g}^{G+} &= \kappa_g^l, \text{ for } g \in \mathcal{G} \text{ such that } Z_{0,g}^G = 1. \\ x_{t,g}^{G+} &= \kappa_g^l, \text{ for } t = 1, \dots, T, \text{ and } g \in \mathcal{G} \text{ such that } (Z_{t-1,g}^G = 0 \cap Z_{t,g}^G = 1). \end{aligned} \quad (1)$$

- The output of any active power generator is bounded by its minimum and maximum capacity. Furthermore, inactive generators (that is, where  $Z_{t,g}^G = 0$ ) must have zero output. Hence we have the following capacity constraints for the set of power generators:

$$\kappa_g^l Z_{t,g}^G \leq x_{t,g}^{G+} \leq \kappa_g^u Z_{t,g}^G, \text{ for } t = 0, \dots, T. \quad (2)$$

- We let the vector  $p_t \in \mathbb{R}^{|\mathcal{N}|}$  represent the nodal power generation at each node in  $\mathcal{N}$ . Given a node  $n_i \in \mathcal{N}$ , we denote with  $\mathcal{G}(n_i)$ ,  $\mathcal{B}(n_i)$ , and  $\mathcal{H}(n_i)$  the sets of respectively generators, storage devices, and wind farms that map to node  $n_i$ . Hence, the  $i$ -th component of  $p_t$  is

$$p_{t,i} = \sum_{g \in \mathcal{G}(n_i)} x_{t,g}^{G+} + \sum_{b \in \mathcal{B}(n_i)} (x_{t,b}^{B+} - x_{t,b}^{B-}) + \sum_{q \in \mathcal{Q}(n_i)} E_{t,q}^W, \text{ } t = 0, \dots, T, \text{ } i = 1, \dots, |\mathcal{N}|. \quad (3)$$

- Since  $y_{t,i}$  denotes the amount of power arriving at node  $n_i \in \mathcal{N}$  at time  $t$ , we can write the electricity demand constraints as

$$y_t + p_t = d_t, \text{ for } t = 0, \dots, T. \quad (4)$$

- Furthermore, we impose flow conservation constraints for storage devices  $b \in \mathcal{B}, t = 0, \dots, T$ :

$$\kappa_b^l \leq R_{t,b}^B + \eta^- x_{t,b}^{B-} - \eta^+ x_{t,b}^{B+} \leq \kappa_b^u. \quad (5)$$

We let  $\mathcal{X}_t$ ,  $t = 0, \dots, T$  be the feasible region defined by the constraints (1)-(5).

In the context of energy storage, complete recourse is an obvious property since we can always choose not to use the storage devices. This is similar to many other real-world applications where a default decision is always available.

In our model, we do not impose ramping constraints for both power generators and the distributed storage devices. While incorporating ramping constraints is relatively easy within the algorithm, it means that we need to include the current output level of each generator in our state variable, which complicates our ability to prove tight bounds.

### The exogenous information process

The only exogenous information processes we consider in this paper is the stochastic change in energy from wind which we model using

$$\hat{E}_t^W = \text{The change in energy from wind between } t-1 \text{ and } t, \quad (6)$$

Energy from wind can be modeled using a rolling forecast of wind provided by an exogenous source (this is how it is done at PJM). These forecasts are provided as exogenous information in the form of a vector

$$f_{tt'}^W = \text{The forecast of wind at time } t' \text{ using the information we have at time } t.$$

Using this notation,  $E_t^W = f_{tt}^W$ , and the exogenous change between  $t$  and  $t+1$  would be

$$\hat{E}_{t+1}^W = f_{t+1,t+1}^W - f_{t,t+1}^W.$$

REMARK 1. In some problems, additional (stochastic) renewable sources such as solar power might be present. In that case we can use similar notation. For example, we can denote with

$$\hat{E}_t^S = \text{The change in energy from solar between } t-1 \text{ and } t.$$

In our numerical work, we view  $L_t$  and  $Z_t^G$  as exogenous information arriving to the system over time. However, we do not model these as exogenous processes, which means we can treat the vectors  $(L_t, Z_t^G)$ ,  $t = 0, \dots, T$  as *latent variables*.

Our wind modeling was derived from a study of off-shore wind reported in Archer et al. (2015), which combined a base set of forecasts from a meteorological model called Weather, Research

and Forecasting (WRF), and a stochastic model of errors in forecasts derived from historical data (and forecasts) provided by PJM. Solar data was derived from actual solar energy (in 5-minute increments) from 23 solar farms operated by PSE&G, a large utility based on New Jersey. We factored up wind and solar data to test our algorithms at high levels of each source of energy.

The model is driven by two other deterministic (but time varying) processes. These are

$$\begin{aligned} L_{ti} &= \text{the load (in MW) at time } t \text{ at node } i, \\ Z_{tg}^G &= \begin{cases} 1, & \text{if generator } g \text{ is on at time } t \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \quad (7)$$

The loads  $L_t$  are those served by PJM in 2010 (in 5-minute increments). The indicator variables  $Z_{tg}^G$  are determined by a unit commitment simulator called SMART-ISO which optimizes fossil generation based on forecasted and actual levels of energy from wind. However, our grid model controls the output level of generators that are already turned on. This way, we trade off generating energy from fossil generators, and storing energy in our grid-level storage devices.

We could introduce other sources of uncertainty such as variations in loads, but our plan is to model high penetrations of wind. At these levels, the uncertainty from wind forecasts is much higher than the uncertainty from other sources. We do not consider outages due to the failure of generators or transmission lines. This model allows us to focus on using storage purely to handle the variability due to wind or solar.

Even though  $L_t$  and  $Z_t^G$  are deterministic processes, we model them as if they are revealed over time. For this reason, we let  $W_t = (E_t^W, L_t, Z_t^G)$  be our exogenous information process. More formally, we are given a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  with a sigma algebra  $\mathcal{F}$ , and a filtration  $\{\emptyset, \Omega\} = \mathcal{F}_1 \subset \mathcal{F}_2 \subset \dots \subset \mathcal{F}_T = \mathcal{F}$ . The stochastic process  $\{W_t\}_{t=1}^T$  is adapted to  $\{\mathcal{F}_t\}_{t=1}^T$ , and the sets of possible realizations of  $W_t$  are denoted with  $\Omega_t$ . Those correspond to nested partitions of  $\Omega$  that are given by  $\{\mathcal{F}_t\}_{t=1}^T$ .

The *post-decision information state*  $I_t^x$  represents all the information in  $S_t^x$  that is not in  $R_t^x$ .  $I_t^x$  depends on the underlying exogenous random process, and contains only the information necessary to model the random transition from the current realization  $\omega_t$  at time  $t$  to the next random realization  $\omega_{t+1}$  at time  $t + 1$ . Hence, in the case of a Markov (lag 1) model,  $I_t^x$  is given by a probability distribution

$$I_t^x = \mathbb{P}(\omega_{t+1} | S_t) = \mathbb{P}_{t+1}(\omega_{t+1} | \omega_t).$$

REMARK 2. Please note that if the given problem features stagewise independence, i.e. a memoryless stochastic process, then all possible realizations  $\omega_t$  would share the same post-decision information state  $I_t^x$ .

### Transition Function

The amount of energy in the storage devices  $R_t^B$  is adjusted to account for injected and withdrawn power,

$$R_{t+1,b}^B = R_{t,b}^{B,x} \quad (8)$$

$$= R_{t,b}^B + \eta_b^- x_{t,b}^{B-} - \eta_t^+ x_{t,b}^{B+}. \quad (9)$$

The amount of power  $E_t^W$  generated by the set of wind farms is adjusted as

$$E_{t+1}^W = E_t^W + \hat{E}_{t+1}^W.$$

The *post-decision resource state*  $R_t^x$  consists of two subvectors  $R_t^{B,x}$  and  $R_t^{G,x}$ . The post-decision amount of energy in the batteries  $R_t^{B,x}$  is equal to the pre-decision amount  $R_t^B$  adjusted for charging, discharging, as well as battery efficiencies,

Moreover, the post-decision state of the generators equals the power generation levels  $x_t^{G+}$ ,

$$R_{t,g}^{G,x} = x_{t,g}^{G+},$$

from which we obtain the next pre-decision state

$$R_{t+1,g}^G = R_{t,g}^{G,x}.$$

### Objective Function

We define the generation costs as the linear functions,

$$\begin{aligned} C(S_t, x_t) &= \langle c_t, x_t \rangle \\ &= \sum_{g \in \mathcal{G}} Z_{t,g}^G c_{t,g}^G x_{t,g}^G + \sum_{b \in \mathcal{B}} c_{t,b}^B x_{t,b}^B. \end{aligned} \quad (10)$$

Our goal is to compute an optimal policy  $X_t^{\pi^*}(S_t)$  that minimizes the total expected generation cost aggregated over the entire time horizon

$$\min_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=0}^T C(S_t, X_t^{\pi}(S_t)) | S_0 \right]. \quad (11)$$

The expectation is taken with respect to a probability measure describing the possible outcomes of the energy from wind. We note that in our energy application, the problem is highly time dependent, which is the reason we have indexed the policy itself by  $t$ , rather than just make it dependent on the state  $S_t$  which depends on time. Our experimental work will focus on modeling a problem over a full daily cycle in 5-minute increments, producing a problem with 288 time periods.

### 3. Stochastic optimization methods

We now contrast two algorithmic strategies for solving these problems which both approach the problem of approximating the value functions in Bellman's equation. The first, widely known as the Stochastic Dual Dynamic Programming (SDDP) within the stochastic programming community was originally introduced in Pereira and Pinto (1991). It uses multidimensional Benders cuts to approximate the value of being in a resource state  $R_t$  using a *sampled* uncertainty model. The second method uses the language and notation of approximate dynamic programming, and approximates the value function using separable, piecewise linear approximations using a full uncertainty model.

These algorithms share the same fundamental style; as we show below, they are both a form of approximate dynamic programming, distinguished primarily by how they approximate the value function, and how they represent the underlying probability space. However, there are several structural differences which make for an interesting comparison, especially in the setting of grid level storage, where the number of storage devices could range from single digits to hundreds, if we wish to anticipate a world of high penetration of renewables and lower cost batteries (which might be in the form of aggregators for electric vehicles).

#### 3.1. From stochastic programming to dynamic programming

We begin by comparing the canonical models for SDDP and ADP. SDDP, with its roots in the stochastic programming community, is a method for solving a problem that is often written in the form

$$\min_{x_0 \in \mathcal{X}_0(S_0)} \langle c_0, x_0 \rangle + \mathbb{E}_1 \left[ \min_{x_1 \in \mathcal{X}_1(S_1)} \langle c_1, x_1 \rangle + \mathbb{E}_2 \left[ \cdots + \mathbb{E}_T \left[ \min_{x_T \in \mathcal{X}_T(S_T)} \langle c_T, x_T \rangle \right] \cdots \right] \right]. \quad (12)$$

where  $\mathcal{X}_0(S_0) = \{x_0 : A_0 x_0 = b_0\}$ , and for  $t \geq 1$ , we define the feasible sets as  $\mathcal{X}_t(S_t) = \{x_t : A_t x_t = b_t - B_{t-1} x_{t-1}, x_t \geq 0\}$ . Here, it is assumed that  $A_t$ ,  $B_t$  and  $b_t$ , as well as the cost vector  $c_t$ , evolve randomly over time and describe the information contained in the stochastic process

$\{W_t\}_{t=1}^T$ , i.e.  $(A_t, B_t, b_t, c_t)$  are  $\mathcal{F}_t$ -measurable matrices and vectors. The stochastic programming community uses several notational styles, but one popular system defines  $\xi_t$  as the new information  $(A_t, B_t, b_t, c_t)$ , and lets  $\xi_{[t]}$  be the history  $\xi_1, \dots, \xi_t$  (see Shapiro et al. (2014)). In this work, we use  $\omega_t$  instead of  $\xi_t$ , and we denote our history by  $\omega_{[t]}$ .

It is common in the stochastic programming literature to write the state variable at time  $t$  as  $(x_{t-1}, \omega_{[t]})$  which captures the dependence of the resource state  $R_t$  on the decision  $x_{t-1}$  (additional random inputs may be contained in  $\omega_t$ ). It is then possible to write a Bellman-style recursion as

$$Q_t(x_{t-1}, \omega_{[t]}) = \min_{x_t} (c_t x_t + \mathbb{E}[Q_{t+1}(x_t, \omega_{[t+1]}) | \omega_{[t]}]). \quad (13)$$

The expectation is computationally intractable, but it is possible to replace it with a series of cuts (Higle and Sen (1991), Pereira and Pinto (1991)), producing the linear program

$$Q_t(x_{t-1}, \omega_{[t]}) = \min_{x_t \in \mathcal{X}_t(x_{t-1}, \omega_{[t]}), v} (c_t x_t + v), \quad (14)$$

where

$$v \geq \alpha_{t+1}^k(\omega_{[t]}) + \beta_{t+1}^k(\omega_{[t]})x_t, \quad \text{for } k = 1, \dots, K, \quad (15)$$

and where  $\mathcal{X}_t$  captures the feasible region for  $x_t$ . Here, equation (15) is generated by solving the dual problem for time  $t+1$ , which means that  $K$  depends on the number of iterations that have been executed. The indexing of the cuts in (15) reflects the fact that we are approximating the value at time  $t+1$ , but it is more accurate to say that it is approximating the recourse function around the post-decision state  $S_t^x$  at time  $t$  (the indexing of time should always reflect the information content of a variable when modeling a stochastic system).

There are two computational issues with using the notational system of writing the state as  $(x_{t-1}, \omega_{[t]})$ . First,  $x_{t-1}$  is generally a very high dimensional vector. In the setting of our energy storage problem, it would have approximately 10,000 variables, one for each transmission line in the PJM grid. Second, indexing on the history  $\omega_{[t]}$  is, of course, problematic. Even if our random variables were scalars (for each time period), the shortest horizon that we are going to consider in our work is 288 time periods (5-minute increments over 24 hours). Retaining a history with more than three or four time periods (even with one variable per time period) is computationally intractable.

Using our notation, we would write

$$\mathbb{E}[Q_{t+1}(x_t, \omega_{[t+1]}) | \omega_{[t]}] = V_t^x(S_t^x) = V_t^x(R_t^x, I_t^x).$$

This allows us to write our Bellman equation in the form of

$$V_t(S_t) = \min_{x_t \in \mathcal{X}_t(S_t)} (c_t x_t + V_t^x(S_t^x))$$

where  $S_t = (R_t, I_t)$  is the pre-decision state, and  $S_t^x = (R_t^x, I_t^x)$  is the post-decision state. Since our problem has a deterministic resource transition process, we know that  $R_{t+1} = R_t^x = B_t x_t$ . We note that the dimension of  $R_t$  and  $R_t^x$  is equal to the number of storage devices, which might be as small as 1 or 10, or as large as 100 (in the experiments that we run). Even a 10-dimensional resource vector would be too large if we were using a lookup table representation, but this is where convexity and Benders cuts allows us to obtain accurate approximations without enumerating the state.

This leaves the problem of the information state. SDDP assumes that the process  $W_t$  is a zeroth order Markov process (widely referred to as intertemporal or interstage independence), which means that  $W_{t+1}$  is independent of  $W_t$ . Under this assumption (which might at least be a reasonable approximation), the post-decision information state  $I_t^x$  is empty and can be ignored, which means that  $V_t^x(S_t^x) = V_t^x(R_t^x)$ . This represents a dramatic simplification of what is otherwise a very high-dimensional ( $x_t$  has 10,000 dimensions) stochastic optimization problem with a long horizon (at least 288 time periods).

A form of the objective function (12) that is more familiar to the stochastic programming community is to create a sampled  $\hat{\Omega} \in \Omega$  and write

$$\min_{x_0, \dots, x_T} \sum_{\omega \in \hat{\Omega}} \sum_{t=0}^T c_t(\omega) x_t(\omega) \quad (16)$$

$$\text{s.t. } A_0 x_0 = b_0 \quad (17)$$

$$B_{t-1}(\omega) x_{t-1}(\omega) + A_t(\omega) x_t(\omega) = b_t(\omega), \quad t = 1, \dots, T, \quad \omega \in \hat{\Omega} \quad (18)$$

$$x_t(\omega) \geq 0, \quad t = 1, \dots, T, \quad \omega \in \hat{\Omega}$$

The notation  $x_t(\omega)$  makes it possible for a decision at time  $t$  to have access to the entire sample path  $\omega$ . For this reason, we have to introduce *nonanticipativity constraints*. This can be done by defining a history  $h_t = (W_1, W_2, \dots, W_t)$ , with the set of histories  $\mathcal{H}_t = \{h_t(\omega), \omega \in \hat{\Omega}\}$ . Next let

$$\hat{\Omega}_t(h_t) = \{\omega \in \hat{\Omega} : (W_1(\omega), \dots, W_t(\omega)) = h_t\},$$

be the set of all sample paths sharing the history  $h_t$ . The nonanticipativity constraints are now given by

$$x_t(h_t) = x_t(\omega), \quad \forall \omega \in \hat{\Omega}_t(h_t), \quad h_t \in \mathcal{H}_t. \quad (19)$$

This is an example of a *sampled model* which is popular in stochastic optimization. The standard approach is to generate  $\hat{\Omega}$  in the form of a scenario tree, where histories are constructed by starting with a history  $h_t$  at time  $t$ , and then branching by sampling realizations of  $W_{t+1}$  given  $h_t$ .

We can make the transition to the formulation used in the dynamic programming community by replacing  $x_t(\omega)$  with  $x_t(h_t)$ , which avoids the need for the nonanticipativity constraints (19). Stochastic programmers will recognize  $h_t$  as a node in the scenario tree. We take this one step further by recognizing that we generally do not need the entire history. Instead, we use the state  $S_t$  which is the minimally dimensioned function of history  $h_t$  which is necessary and sufficient (along with the exogenous information) to model our system from time  $t$  onward. We then have to choose  $x_t(S_t)$  that satisfies the constraints  $A_t x_t = b_t - B_{t-1} x_{t-1} = R_t$ .

The dynamic programming community approaches the problem by recognizing that  $x_t(S_t)$  is a function called a policy, that we write  $X_t^\pi(S_t)$ . The objective function can now be written

$$\min_{\pi} \mathbb{E} \left[ \sum_{t=0}^T C(S_t, X_t^\pi(S_t)) | S_0 \right]. \quad (20)$$

The dynamics of the system are captured through the transition function

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}) \quad (21)$$

where  $x_t = X_t^\pi(S_t)$ . We note that if we fix a policy, we can simulate the value of the policy for  $\omega \in \Omega$  using

$$v^\pi(\omega) = \sum_{t=0}^T C(S_t(\omega), X_t^\pi(S_t(\omega))) \quad (22)$$

where  $S_{t+1}(\omega) = S^M(S_t(\omega), X_t^\pi(S_t(\omega)), W_{t+1}(\omega))$ . We note that while simulating the policy in (22), it is quite easy to make the outcome  $W_{t+1}(\omega)$  dependent on both  $S_t$  and  $x_t$ , which can be a valuable feature in energy applications (discharging energy into the grid can dampen electricity prices). The decisions  $x_t = X_t^\pi(S_t)$  must satisfy  $x_t \in \mathcal{X}_t = \{x_t : A_t x_t = b_t - B_{t-1} x_{t-1}, x_t \geq 0\}$ . Nonanticipativity is satisfied automatically by writing the policy as dependent on the state rather than the sample path.



The dynamic programming community often skips equation (20) and goes directly to Bellman's equation, which would be written

$$V_t(S_t) = \min_{x \in \mathcal{X}_t} (C(S_t, x) + \mathbb{E}[V_{t+1}(S_{t+1})|S_t]). \quad (23)$$

Exploiting the post-decision state allows us to drop the expectation, giving us

$$V_t(S_t) = \min_{x \in \mathcal{X}_t} (C(S_t, x) + V_t^x(S_t^x)). \quad (24)$$

If we make the same assumption that the information process  $\{W_t\}_{t=1}^T$  is independent over time, then  $S_t^x = R_t^x = B_t x_t$ . Exploiting the convexity of  $V_t^x(R_t^x)$  (as well as  $V_t(R_t, I_t)$ ), we can approximate  $V_t^x(S_t^x) = V_t^x(R_t^x)$  using Benders cuts (as we did in equations (14)-(15)), but we can also experiment with other approximations. Below, we test the idea of approximating the value function using separable, piecewise linear functions which we can write

$$V_t^x(R_t^x) \approx \sum_{i \in \mathcal{I}} \bar{V}_{ti}^x(R_{ti}^x), \quad (25)$$

where  $\bar{V}_{ti}^x(R_{ti}^x)$  is a one-dimensional piecewise linear and convex function. These approximations can be estimated from dual variables from the sampled problem solved at time  $t + 1$  using algorithms such as CAVE and SPAR (see Powell (2011)[Chapter 13], or Godfrey and Powell (2001) and Powell et al. (2004)).

We note that while we can assume that the post-decision information state  $I_t^x$  is empty, there may be applications where the future depends on a compact information state. For our energy application, we might characterize the weather using a small number of states that capture temperature and the likelihood of precipitation. If we can represent these “states of weather” using perhaps 10 or 20 values, then we can create indexed convex approximations using either Benders cuts or separable approximations. Such models have been described as “Markov” in the literature (for some reason, many authors assume that a “Markov” model state space has to be small and discrete). Löhndorf et al. (2013) and Lohndorf and Wozabal (2015) pursue this idea under the name “Approximate dual dynamic programming,” but an alternative name is “Markov SDDP” which would help to communicate the method to the stochastic programming community.

We are now going to present and test two algorithms that are structurally very similar. Recognizing that the fundamental structure of these two algorithms is quite similar, we undertake a series of qualitative and empirical comparisons that highlight the two key differences: how we model the information process (sampled or full), and how we approximate the value function (multidimensional Benders cuts or piecewise linear, separable).

### 3.2. SDDP with Benders cuts

Stochastic dual dynamic programming (SDDP) has long enjoyed special attention from the stochastic programming community. The algorithm is summarized in the online supplement. Key features of the algorithm include:

- The algorithm solves a sampled version of the problem, using  $\hat{\Omega}_t$  that are chosen before the algorithm starts.
- The value functions are approximated by multidimensional Benders cuts.
- It uses a forward pass, simulating the process of making decisions by solving a sequence of linear programs given by (14)-(15). Regularization is used to stabilize the solution.
- Dual solutions are computed for *each* sample realization in  $\hat{\Omega}_t$ , computed in a backward pass. A new cut is created by averaging across the duals using the sample  $\hat{\Omega}_t$ .

### 3.3. ADP-SPWL with separable, piecewise linear value function approximation

The algorithm ADP-SPWL is described in the online supplement. Key characteristics of the algorithm include:

- The value functions are approximated using separable, piecewise linear value functions.
- It uses a forward pass, simulating the process of making decisions by solving a sequence of linear programs given by (24)-(25). Samples at time  $t$  are generated on the fly from the full sample space  $\Omega_t$ , which means that they can reflect the state  $S_t$ .
- Numerical derivatives (requiring the solution of a linear program) are computed for *each* storage device. These derivatives are then smoothed to create updated VFAs for each storage device.

We note that our algorithm could have been implemented as a pure forward pass procedure, as has been done in transportation applications (see e.g., Topaloglu and Powell (2006a)). In these applications, computing numerical derivatives for each storage device would not be necessary. However, a pure forward pass algorithm can exhibit slow convergence when decisions at one point in time have an impact many time periods in the future, which is the case in our energy storage application. Our battery storage example models a day (or more) in 5-minute increments. We may, for example, wish to store energy at 3 pm to use at 9 pm, which is 72 time periods in the future. For this type of problem, a backward pass dramatically accelerates the learning over time. However, this means that we need to know the flow augmenting path from an increment of energy in a battery at time  $t$  in terms of how it impacts the resource state  $R_{t+1}$ . This is the reason that numerical derivatives are necessary.

### 3.4. A comparison

Stochastic dual dynamic programming	Separable piece-wise linear value functions
A fixed sample is generated once and used for all iterations (the sampled model).	New samples are drawn each iteration from the original information model.
Solves a subproblem for each random realization $\omega_t \in \hat{\Omega}_t$ at each time $t = 0, \dots, T$ .	Solves a subproblem for each post-decision resource dimension $R_{t,m}^x$ at each time $t = 0, \dots, T$ .
Multidimensional Benders cuts for VFAs.	Separable, piecewise linear VFAs.
Growing set of hyperplanes.	Growing set of kinks in each VFA.
Lower bounds for sampled model.	No lower bounds.
Stochastic process must be independent of the state.	Sampled outcomes may depend on the state.

Table 3.4 shows a side-by-side comparison of characteristics of SDDP and the ADP algorithm using a separable, piecewise linear value function. There are several differences which should be highlighted.

- SDDP requires the use of a sampled model because it is averaging multidimensional cuts. The likelihood of visiting the same multidimensional resource state  $R_t$  on two successive iterations is nearly zero. By contrast, ADP-SPWL exploits the use of separable approximations, which allows us to smooth observations of slopes from different sample realizations into a single function. ADP-SPWL updates the marginal value functions for all of the resource dimensions at each time period, in each backward pass.
- By generating multidimensional cuts, SDDP enjoys the feature that it can generate upper bounds on the solution, which can be used to help evaluate the quality of the solution. However, for our energy storage application, it is important to use care when interpreting these bounds, because the objective function features a large constant term representing the value of a myopic policy (setting the value functions equal to zero). When evaluating solution quality using upper and lower bounds, the gap can seem small if we do not subtract this constant term.
- SDDP requires solving a linear program for each sample realization at each time period. ADP-SPWL requires solving a linear program for each dimension of our resource state variable.
- Since SDDP is solving a sampled approximation, the bounds that it generates are, strictly speaking, bounds on the optimal solution of the sampled problem.

- SDDP must use a sampled model that is generated before the algorithm starts. ADP-SPWL draws samples dynamically during the forward pass, making it possible to generate samples that depend on the state  $S_t$  and/or the action  $x_t$ .
- Benders decomposition is known to show slow convergence as the dimensionality of the resource state grows, but the impact of dimensionality will depend on the characteristics of the problem. The separable approximations for ADP-SPWL has been shown to work well at high dimensions, but this work is based on experiments in transportation and logistics where separability is likely to be a better approximations (see Topaloglu and Powell (2006a) and Bouzaïene-Ayari et al. (2014) for illustrations). In our grid application, substitution of energy between storage devices is much easier, suggesting that a separable approximation may not work as well.
- Neither algorithm has been tested against serious competition. SDDP has been evaluated purely on the basis of its bounds on the sampled approximation. ADP-SPWL has been evaluated primarily through comparisons against optimal solutions on deterministic problems (see Topaloglu and Powell (2006b)) or scalar, stochastic problems Jiang et al. (2014).

#### 4. Computational comparisons of SDDP and ADP-SPWL

In this section we study the computational performance of the algorithms proposed above. We introduce the following simplifications to the state space:

- The post-decision resource space is collapsed around  $R_t^B$ . In this way, we ignore rampup limits for the sake of numerical testing.
- The information space is collapsed around  $E_t^W$  only since that is the main source of volatility.

In our numerical experiments we focus our analysis on the following questions:

- How is the computational performance of SDDP and ADP-SPWL affected by:
  - the dimension of the resource vector  $R_t^x$ ?
  - the size of the sample sets  $|\hat{\Omega}_t|$ ?

Our experimental work was conducted using the setting of optimizing grid level storage for a large transmission grid managed by PJM Interconnection. PJM manages grid level storage devices from a single location, making it a natural setting for testing our algorithms. As of this writing, grid level storage is dropping in price, providing a meaningful setting to evaluate the performance of our algorithms for a wide range of storage devices, challenging the ability of the algorithms to handle high dimensional applications. For this reason, we conducted tests on networks with

up to 100 storage devices. These are much higher dimensional problems than prior research that has focused on the management of water reservoirs. In order to be able to use high-dimensional Benders approximations, we consider a quadratic regularization extension of SDDP that was originally introduced by Asamov and Powell (2015a). The algorithm was implemented in Java, and the IBM ILOG CPLEX 12.4 solver was used for the solution of both linear and quadratic convex optimization problems. In addition, the relative complementarity tolerance of CPLEX was set to  $10^{-12}$ .

Another distinguishing feature of our grid storage setting (compared to prior experimental work) is that a natural time step is 5 minutes, which is the frequency with which real-time electricity prices (known as LMPs, for locational marginal prices) are updated on the PJM grid. We anticipate using storage devices to hold energy over horizons of several hours. For this reason, we used a 24 hour model, divided into 5-minute increments, for 288 time periods, which is quite large compared to many applications using this algorithmic technology.

Below we describe the construction of the network, the representation of the exogenous stochastic process, and finally we present the results of an extensive set of experiments investigating the effect of regularization, the number of storage devices (which determines the dimensionality of  $R_t^x$ ), and the presence of an exogenous post-decision information state, on the rate of convergence and solution quality.

#### 4.1. The network

We performed our experiments using an aggregated version of the PJM grid. Instead of the full network with 9,000 buses and 14,000 transmission lines, we limited our analysis to the higher voltage lines, producing a grid with 1,360 buses and 1,715 transmission lines. Off-shore wind power was simulated for a set of hypothetical wind turbines with a combined maximum capacity of 16 GW. Moreover, we consider a daily time horizon with 5-minute discretization resulting in a total of 288 time periods.

The data was prepared by first running a unit-commitment simulator called SMART-ISO that determines which generators are on or off at each point in time, given forecasts of wind generated from a planned set of off-shore wind farms. We made the assumption that the use of grid level storage would not change which generators are on or off at any point in time. However, we simultaneously optimize the generator output levels, while charging and discharging of storage devices around the grid in the presence of stochastic injections from the wind farms.

We placed the distributed storage devices at the points-of-interconnection for wind farms, as well as the buses with the highest demand. Each storage device is characterized by its minimum and maximum energy capacity, its charging and discharging efficiency, and its variable storage cost. The control of multiple storage devices in a distributed energy system is a challenging task that depends on a variety of factors such as the location of each device, and the presence of transmission line congestion. A good storage algorithm needs to respond to daily variations in supply, demand and congestion, taking advantage of opportunities to store energy near generation points (to avoid congestion) or near load points (during off-peak periods). It has to balance when and where to store and discharge in a stochastic, time-dependent setting, providing a challenging test environment for our algorithm.

#### 4.2. The exogenous information

Our only source of uncertainty (the exogenous information) was from the injected wind from the offshore wind farms. In order to calibrate our stochastic wind error model, we employed historical wind data and speed measurements of off-shore wind for the month of January 2010. For each time period  $t$ , we consider a set  $\hat{\Omega}_t$  of vectors of possible wind speed realizations which correspond to  $|\hat{\Omega}_t|$  different weather regimes. Plots of simulated wind power at a given wind farm can be seen in Figure 2 in the online supplement.

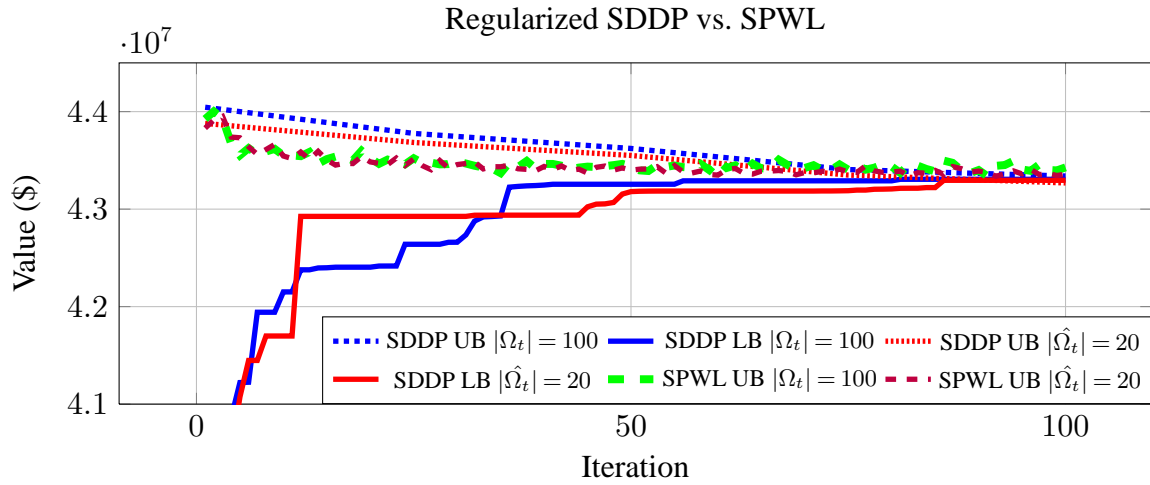
#### 4.3. Deterministic Experiments

In the online supplement we present deterministic experiments that allow us to perform a side by side comparisons of objective values and decisions made by SDDP and ADP-SPWL. We can see that in the instance with only five storage devices, the SDDP solution is closer to the optimal than the solution of ADP-SPWL. However, as the dimension of the post-decision value functions increases, the separable value function approximations outperform their Benders counterparts in terms of both objective value and solution quality. Still, the real test has to be on a stochastic dataset, since a deterministic problem allows the algorithms to learn the states of other storage devices at each point in time.

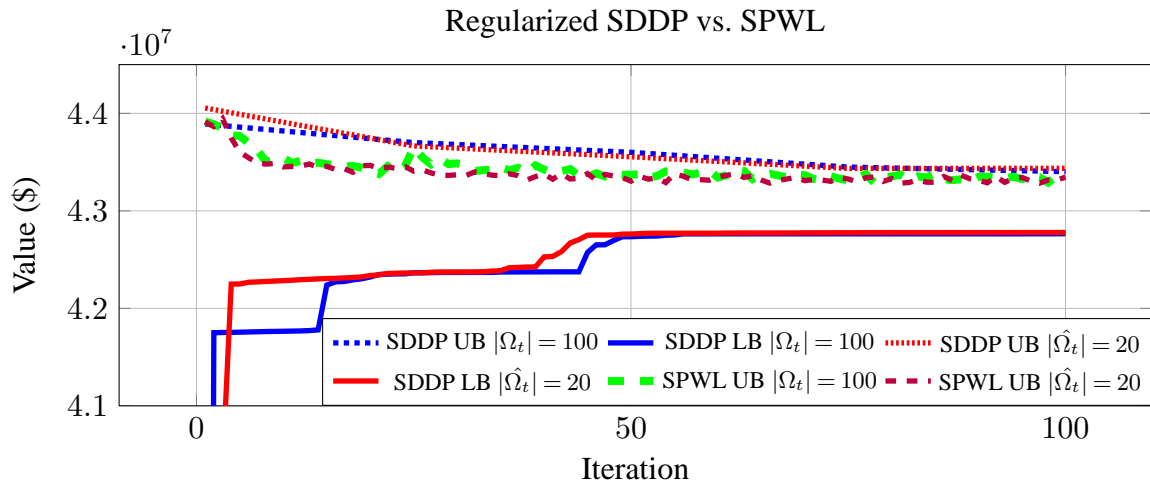
#### 4.4. Stochastic Experiments

In this section, we turn our attention to stochastic experiments, and at the same time address issues related to the use of a sampled model by SDDP. The first question that we study is the computational effect of increasing the dimensionality of  $R_t^x$ . This is a major issue, largely overlooked in the SDDP community. SDDP has always been presented as a way of circumventing the curse of

dimensionality, but in practice it has been used only for the solution of instances with very low dimensional value functions. Thus, the question of its practical applicability to high-dimensional problems has been left unanswered until now. The plots below illustrate several important points about solving such large-scale stochastic problems. First, both the regularized version of SDDP, as well as ADP-SPWL seem applicable for instances with large resource dimensions  $|R_t^x|$  that would be intractable for non-regularized Benders methods (even in the deterministic case). The results suggest that SDDP regularization allows practitioners to consistently obtain high quality solutions within approximately 50 iterations for all of the given problems. However, during initial iterations the ADP-SPWL approach can exhibit even faster convergence than the regularized SDDP.



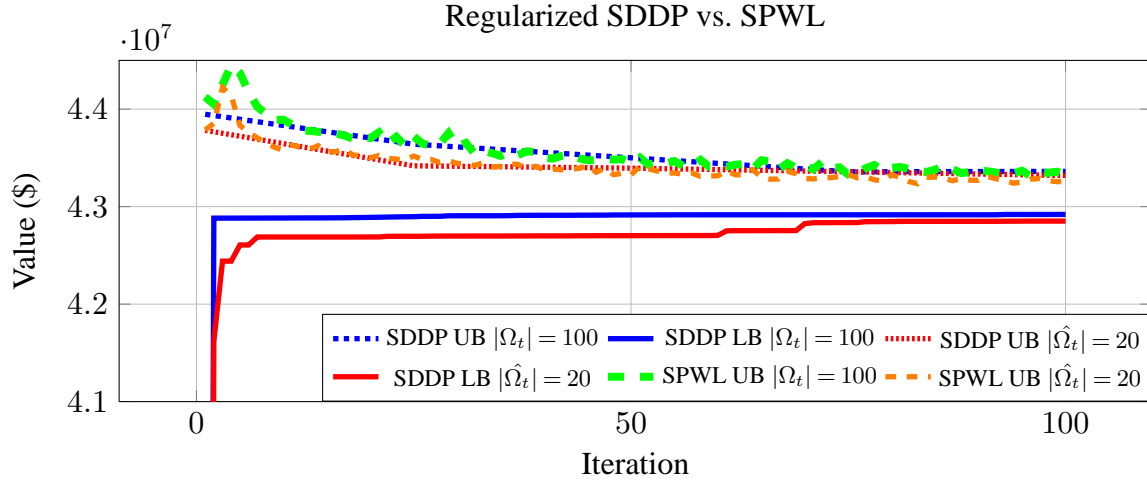
**Figure 1** Numerical comparison of multistage stochastic optimization methods for  $|R_t^x| = 25$ .



**Figure 2** Numerical comparison of multistage stochastic optimization methods for  $|R_t^x| = 50$ .

<div style="display: flex; align-items: center;"> <div style="border-bottom: 1px solid black; padding-right: 10px;"> <math>( R_t^x ,  \hat{\Omega}_t )</math> </div> <div style="border-bottom: 1px solid black; padding-left: 10px;">Method</div> </div>		# Iterations	1	50	100
(25, 50)	SDDP (Regularization)		712.0	804.4	902.1
	ADP - SPWL		284.0	310.2	325.0
(25, 20)	SDDP (Regularization)		176.0	180.7	185.4
	ADP - SPLW		116.0	136.4	147.1
(50, 50)	SDDP (Regularization)		763.0	869.1	986.4
	ADP - SPWL		719.0	762.1	795.8
(50, 50)	SDDP (Regularization)		451.0	487.4	517.6
	ADP - SPWL		975.0	1081.9	964.5
(100, 100)	SDDP (Regularization)		2229.0	2657.1	3030.8
	ADP - SPWL		3167.0	3016.0	2992.7
(100, 20)	SDDP (Regularization)		478.0	616.9	683.9
	ADP - SPWL		3167.0	3016.0	2992.7

**Table 1** Computational time per iteration (in seconds) for stochastic optimization methods.



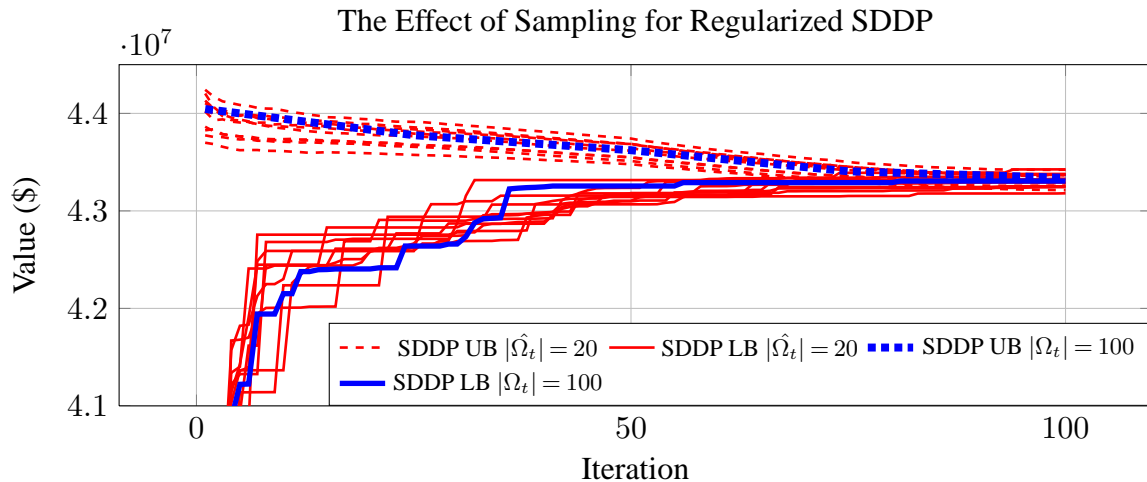
**Figure 3** Numerical comparison of multistage stochastic optimization methods for  $|R_t^x| = 100$ .

Table 1 shows the CPU times (in seconds) per iteration for problems with up to 100 storage devices, and different sample sizes  $\hat{\Omega}_t$ . We can see that computational times of all methods depend on the choice of problem parameters, and there is not a single method that always outperforms the competition. Hence, the choice of the solution method should be made on a case-by-case basis taking into account factors such as preferred solution structure, limitations on computational time

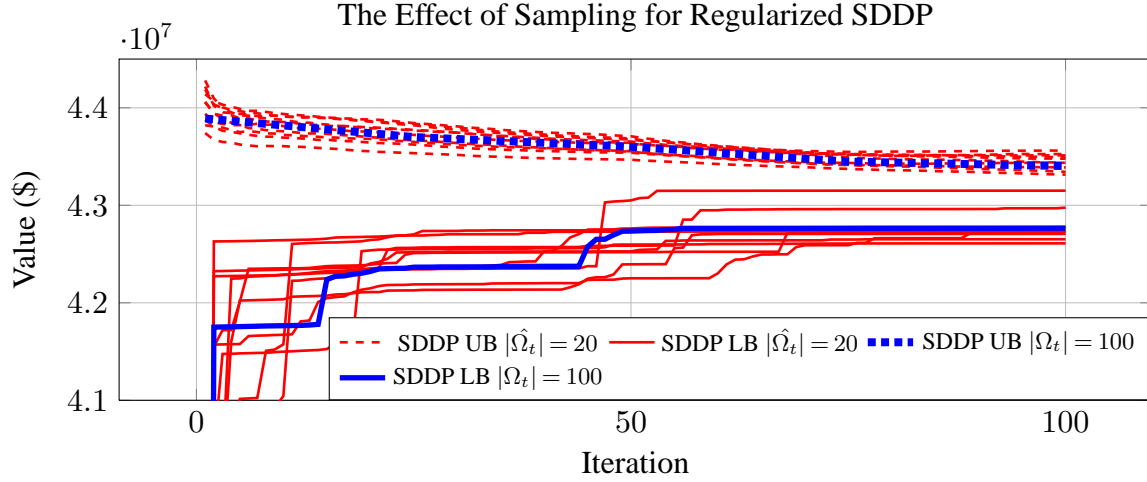


and resources, ease of implementation, the characteristics of the stochastic process, the quality of the available linear and quadratic programming solvers, and potential future needs.

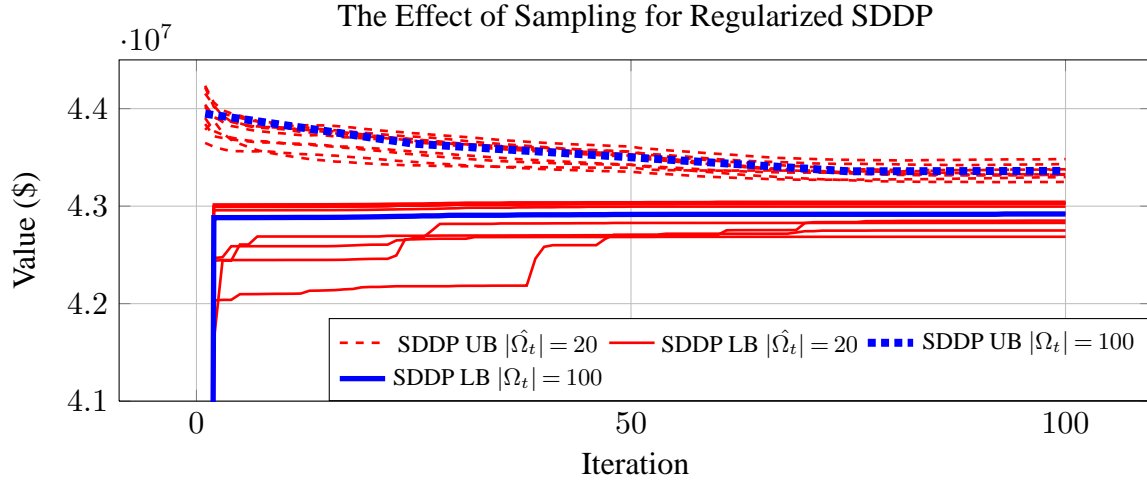
In addition, we are also interested in the magnitude of the errors arising from the use of a sampled model. In order to gain estimates, we consider runs with a sample sizes  $|\Omega_t| = 100$ , and then do smaller samples  $\hat{\Omega}_t$  drawn from  $\Omega_t$ , and compare the resulting objective values. The use of a small sample  $\hat{\Omega}_t$  could lead to a major computational speed-up since SDDP-type methods need to solve an optimization problem for each random realization at each time step in the backward pass of every iteration. However, we can also see that the use of a small sample  $\hat{\Omega}_t$  can also introduce significant approximation errors. It is widely believed (and theoretically expected) that a smaller sample size should result in a lower optimal objective value due to overfitting. However, we can see that in practice that is not necessarily the case since we rarely solve the problem to optimality. Instead a smaller sample can result in both underestimated, as well as overestimated bounds and there is no obvious criterion to help us distinguish between the two cases a priori. In addition, it is known that the square root law applies to the optimal values of the sampled optimization problems (i.e. in order to gain one decimal place of precision, we need to increase the sample size by a factor of 100) but large-scale real world problems are rarely solved to optimality.



**Figure 4** Numerical comparison of multistage stochastic optimization methods for  $|R_t^x| = 25$ .



**Figure 5** Numerical comparison of multistage stochastic optimization methods for  $|R_t^x| = 50$ .



**Figure 6** Numerical comparison of multistage stochastic optimization methods for  $|R_t^x| = 100$ .

Moreover, practitioners usually use a small fixed number of iterations (hundreds or thousands) and hence there is no direct way to determine how a smaller sample size would affect the resulting policy and optimality gap. Finally, we would like to emphasize that typically sampling is not an issue for approximate dynamic programming with separable value functions since it works directly with the full probability model, and hence it avoids sampling errors altogether. Unfortunately, currently there does not exist any approach that would allow us to extend this property to the SDDP framework.

## 5. Conclusion

Multistage stochastic optimization problems with long time horizons appear in various fields and the computational solution of such models is a topic of growing importance. In our work we have

compared the performance of SDDP (with regularization) and ADP-SPWL for the solution of multistage stochastic problems in energy storage.

On one hand, approximations with separable value functions feature many desirable properties. For instance, the random process can depend on the decisions made at each time step, and use of a wide variety of probability models is possible (even computer simulations). However, such broad applicability comes at a price. In general, practitioners do not have any guarantees for the quality of policies derived with separable value functions since lower bounds are not readily available.

When optimality guarantees are needed, researchers often resort to the SDDP framework which requires a memoryless stochastic process that is exogenous to the decisions made by the model. Moreover, SDDP employs a finite sample to construct a sampled average approximation to the original problem. When the given formulation is convex, lower bounds are readily available and an optimality bound for the policy of the sampled problem can be estimated. However, our numerical work indicates that the relationship between the bounds derived from the sampled model, and the bounds derived from the full model cannot be determined a priori.

In general, the practical application of SDDP has been limited to problems with low-dimensional value functions such as hydro-power problems with a small number of (groups of) reservoirs. In our experiments we show that such a limitation can be overcome with the use of a regularization technique proposed by Asamov and Powell (2015a), and approximations with Benders cuts can compete with ADP methods with separable value functions on larger problems than previously known. Moreover, we have studied the resulting policies and compared them to the optimal solutions in deterministic instances. Our results suggest that neither algorithm is universally best, and we recommend that practitioners choose the solution method depending on the characteristics of the problem at hand.

In the future we plan to extend the current work to numerical testing of non-linear problems, including risk-averse models formulated as time-consistent compositions of coherent measures of risk (Asamov and Ruszczyński (2014)), as well as problems with multiple objectives (Young et al. (2010)).

## References

- Archer, C. L., H. P. Simao, W. Kempton, W. B. Powell, and M. J. Dvorak (2015). The challenge of integrating offshore wind power in the U.S. electric grid. Part I: Wind forecast error.
- Asamov, T. and W. B. Powell (2015a). Regularized decomposition of high-dimensional multistage stochastic programs with markov uncertainty. *arXiv preprint arXiv:1505.02227*.

- Asamov, T. and W. B. Powell (2015b). Regularized Decomposition of High Dimensional Multistage Stochastic Programs with Markov Uncertainty. Technical report, Princeton University, Princeton, N.J.
- Asamov, T. and A. Ruszczyński (2014). Time-consistent approximations of risk-averse multistage stochastic optimization problems. *Mathematical Programming* pp. 1–35. ISSN 0025-5610.  
URL <http://dx.doi.org/10.1007/s10107-014-0813-x>
- Birge, J. R. (1985). Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs. *Operations Research* 33(5), 989–1007. ISSN 0030-364X.
- Bouzaïene-Ayari, B., C. Cheng, S. Das, R. Fiorillo, and W. B. Powell (2014). From Single Commodity to Multiattribute Models for Locomotive Optimization : A Comparison of Optimal Integer Programming and Approximate Dynamic Programming. *Transportation Science* pp. 1–24.
- Girardeau, P. and A. Philpott (2015). On the convergence of decomposition methods for multi-stage stochastic convex programs. *Mathematics of Operations Research* 40(1), 130–145.  
URL [http://www.optimization-online.org/DB\\_FILE/2012/04/3445.pdf](http://www.optimization-online.org/DB_FILE/2012/04/3445.pdf)
- Godfrey, G. and W. B. Powell (2001). An Adaptive, Distribution-Free Algorithm for the Newsvendor Problem with Censored Demands, with Applications to Inventory and Distribution. *Management Science* 47(8), 1101–1112. ISSN 0025-1909.  
URL <http://mansci.journal.informs.org/cgi/doi/10.1287/mnsc.47.8.1101.10231>
- Higle, J. and S. Sen (1991). Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research* 16(3), 650–669.  
URL <http://www.jstor.org/stable/3690044>
- Jiang, D. R., T. V. Pham, W. B. Powell, D. F. Salas, and W. R. Scott (2014). A comparison of approximate dynamic programming techniques on benchmark energy storage problems: Does anything work? In *IEEE Conference on Approximate Dynamic Programming and Reinforcement Learning*, pp. 1–8. IEEE, Orlando, FL.
- Linowsky, K. and A. B. Philpott (2005). On the Convergence of Sampling-Based Decomposition Algorithms for Multistage Stochastic Programs. *Journal of Optimization Theory and Applications* 125(2), 349–366. ISSN 0022-3239.  
URL <http://www.springerlink.com/index/10.1007/s10957-004-1842-z>
- Lohndorf, N. and D. Wozabal (2015). Optimal gas storage valuation and futures trading under a high-dimensional price process. Technical report.
- Löhndorf, N., D. Wozabal, and S. Minner (2013). Optimizing Trading Decisions for Hydro Storage Systems Using Approximate Dual Dynamic Programming. *Operation Research* 61(4), 810–823.
- Lohndorf, N., D. Wozabal, and S. Minner (2013). Optimizing Trading Decisions for Hydro Storage Systems Using Approximate Dual Dynamic Programming Optimizing Trading Decisions for Hydro Storage Systems Using Approximate Dual Dynamic Programming. *Operation Research* 61(4), 810–823.

- 
- Mo, B. and A. Gjelsvik (2001). Integrated Risk Management of Hydro Power Scheduling and Contract Management. *Power* 16(2), 216–221.
- Pereira, M. and L. M. Pinto (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming* 52(1-3), 359–375.
- Philpott, A. B. and Z. Guan (2008). On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters* 36(4), 450–455. ISSN 01676377.  
URL <http://linkinghub.elsevier.com/retrieve/pii/S0167637708000308>
- Pinto, G. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming* 52, 359–375.
- Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, Hoboken, NJ, 2 edition.
- Powell, W. B., A. Ruszczycki, and H. Topaloglu (2004). Learning Algorithms for Separable Approximations of Discrete Stochastic Optimization Problems. *Mathematics of Operations Research* 29(4), 814–836. ISSN 0364-765X.  
URL <http://mor.journal.informs.org/cgi/doi/10.1287/moor.1040.0107>
- Salas, D. F. and W. B. Powell (2015). Benchmarking a Scalable Approximate Dynamic Programming Algorithm for Stochastic Control of Multidimensional Energy Storage Problems. *Inform. J. on Computing* pp. 1–41.  
URL <http://www.castlelab.princeton.edu/Papers/SalasPowell-BenchmarkingADPformultidimens>
- Sen, S. and Z. Zhou (2014). Multistage Stochastic Decomposition: A Bridge Between Stochastic Programming and Approximate Dynamic Programming. *SIAM J. Optimization* 24(1), 127–153.
- Shapiro, A. (2011). Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research* 209(1), 63–72. ISSN 03772217.
- Shapiro, A., D. Dentcheva, and A. Ruszczycki (2014). *Lectures on Stochastic Programming: Modeling and theory*. SIAM, Philadelphia, 2 edition.
- Simão, H. P., W. B. Powell, C. L. Archer, and W. Kempton (2015). The challenge of integrating offshore wind power in the US electric grid . Part II : Simulation of the PJM market operation . Technical report, Princeton University, Princeton, N.J.
- Topaloglu, H. and W. B. Powell (2006a). Dynamic-Programming Approximations for Stochastic Time-Staged Integer Multicommodity-Flow Problems. *INFORMS Journal on Computing* 18(1), 31–42. ISSN 1091-9856.
- Topaloglu, H. and W. B. Powell (2006b). Dynamic Programming Approximations for Stochastic, Time-Staged Integer Multicommodity Flow Problems. *Inform. Journal on Computing* 18, 31–42.
- Young, C. M., M. Li, Y. Zhu, M. Xie, E. A. Elsayed, and T. Asamov (2010). Multiobjective optimization of a port-of-entry inspection policy. *Automation Science and Engineering, IEEE Transactions on* 7(2), 392–400.